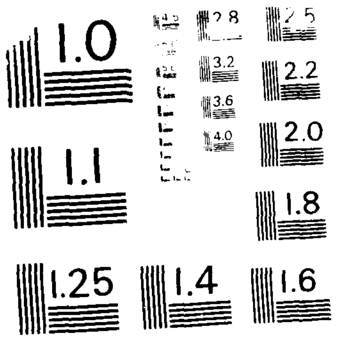

1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

AFIT/GST/ENS/88M-9

SEARCH- AN INTERACTIVE COMPUTER PROGRAM FOR OPTIMIZING TWO-VARIABLE, UNCONSTRAINED EXPERIMENTS OR SIMULATION

THESIS

Billy G. Ploetner
Major, USAF

AFIT/GST/ENS/88M-9

Approved for public release; distribution unlimited

AFIT/GST/ENS/88M-9

SEARCH: AN INTERACTIVE COMPUTER PROGRAM FOR OPTIMIZING TWO-VARIABLE, UNCONSTRAINED EXPERIMENTS OR SIMULATION

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Operations Research

Billy G. Ploetner
Major, USAF

March 1988

Preface

The purpose of this study was to develop a computer program that incorporates the most efficient techniques for solving optimization problems for simulation and experimental test.

This research effort could not have been accomplished without a great deal of help from others. Foremost, I am indebted to my advisor Lt. Col. Joseph Faix for his expertise on the subject and his enduring patience. I wish to thank Major Joseph Litko and Lt. Col. William Rowell for their advice and guidance. I also wish to thank those fellow classmates of mine who have provided assistance throughout the duration of this research effort. Finally, I wish to thank my wife Charlotte and my two sons Marty and Todd for their understanding, concern, and sacrifice.

Billy G. Flietner

Table of Contents

List of Figures

List of Tables

AFIT/GST/ENS/88M-9

Abstract

The purpose of this study was to develop a computer program that incorporates efficient techniques for solving optimization of experimental test or simulation models. The program is interactive and user-friendly. The program is written in Fortran but can be attach to any simulation model or experiment. The program is limited to two independent variables and one dependent variable. The algorithm of the main program is steepest ascent partan.

The study compared several gradient methods and found 2-k factorial the most efficient. The study also concluded that Davies, Swann, and Campey (DSC)- Powell was the most useful line search. The study uses an improvement by Faix to the PARTAN method to eliminate the final line search. The program is designed to efficiently solve second-order equations and less efficiently higher order equations.

SEARCH: AN INTERACTIVE COMPUTER PROGRAM FOR OPTIMIZING TWO-VARIABLE, UNCONSTRAINED EXPERIMENTS OR SIMULATION

I. Introduction

General Background

In science, industry, military operations, business and most facets of life, countless effort is spent trying to improve "production" and maximize one's output. For some examples, in chemistry, what amount of chemical A mixed with chemical B produces the largest amount of product C. In military operations, how many and of what type weapon should attack which targets to optimize the damage expectancy. In business, how many of a certain item type should be ordered to minimize storage while maximizing sales. Or, in one's personal finance, what amount should be paid on certain debts and what amounts should be invested in which type of savings to maximize one's wealth. So, whether it is the right amount of an input that produces the best output or the correct number of inputs to provide the best results, optimization is a problem dealt with almost daily. Hillier and Lieberman point out, "this 'search for optimality' is a very important theme in operations research"(3:5).

Operations research covers a broad spectrum of optimization programming. At one end lies the well developed linear programming with its maximization of a linear objective function and restrictions by linear constraints. Going down the spectrum in limitations, one finds the lesser developed non-linear programs such as quadratic, geometric, and fractional programming. These algorithms still optimize an objective function, but the restrictions of linearity are relaxed. Still further down this restrictive ladder, one comes to direct search methods. Here, optimization of the problem is still the goal, but the formula of the objective function is unknown. Direct search techniques use experiments (trial and error) to gain information about the optimum. This method involves either experimenting with the real system itself or experimenting with a simulation model of the system. This search category includes techniques such as exhaustive search, random search, and direct search.

Wilde clarifies this category of problems as follows:

> The search problem is to find, after only a few experiments, a set of operating conditions yielding a value of the criterion y which is close to the best attainable. From another point of view, the problem is to reach a specified minimum acceptable level of performance in as few trials as possible. Geometrically speaking, we would like to climb up the response surface as quickly as possible, even though the only information we have about the surface comes from the past experiments we have run [9:64].

New and better search techniques are evolving as the research for more efficient methods is continued. The thrust of this study and criterion of effectiveness of methods is efficiency in reaching the optimum. More efficient is defined as requiring fewer experiments to reach the vicinity of the optimum. Thus, better, more efficient methods arrive at the optimum using fewer experiments or simulations.

The simplest, and probably first, search technique used was the exhaustive search, or mere enumeration. This technique involves looking at all possible combinations of input variables and selecting the combination giving the highest output. This accurate method probably saw a short rebirth with the advancements of computers. The exhaustive method works fine for small problems; however, even with computers, a more efficient method is needed to save time and cost.

Since with exhaustive search it might often times be prohibitive, random search might be used. This involves randomly selecting input combinations for testing. The problem with random search is never knowing when the optimum has been reached unless all points are tested. If only a few experiments are possible, random search might be considered the best choice for a large problem.

From the need for a more scientific method, the direct search methods were developed. Direct search is a planned, mathematical search that leads one to the optimum. Over the last thirty years there have been numerous direct search plans developed. Search plans basically fall into two categories: simultaneous and sequential. Plans specifying the location of every experiment before any results are known are called simultaneous, while a plan permitting the experimenter to base future experiments on past outcomes are called sequential (8:5). Simultaneous search plans, usually called experimental designs, have been developed that systematically test points in a specified region of interest. Response surface methodology (RSM) takes the experimental design and calculates an estimated equation of the real system from which an expected optimum can be derived mathematically. Numerous sequential search algorithms have also been developed. These algorithms generally entail the use of gradients, directional line searches, geometry, and sometimes curve fitting.

Problem Statement

Frequently, engineers are given problems to solve in which they want the optimal solution (either maximization or minimization) and no equations or formulas exist of the objective function. Thus experimenting (or simulating) provides the only clues for the location of the optimum.

There is a need for a computer software program that guides a person to the optimum whether maximum or minimum when only simulation or experimentation is available. It should combine various efficient direct search techniques in an algorithm to expedite the search. It should employ simple techniques that a practicing engineer should understand. It's goal should be to minimize the number of experimental tests (simulations) required. Basically, it should be simple for the user to implement and use.

Research Objectives

The overall objective of this research is to develop an interactive, user-friendly computer package which allows one to find the optimal response to experimental test or simulation models. The program will contain the most efficient search techniques. The program will quickly solve for the optimum for quadratic surfaces and many higher order equations.

Subobjectives of this research effort are as follows:

(1) The efficiency of different techniques during different phases of the program compared in order to select the most efficient techniques for the program. The measure of effectiveness for efficiency being the least experimental trials needed for the required accuracy.

(2) A verification of the computer program accomplished showing that it does solve optimization problems. This would entail taking various problems and checking to see if the program can find the optimal solution.

Scope

The limitations of this research are:

(1) The independent (input) variables are limited to two and they are continuous real variables.

(2) There is a single dependent (output) variable from the model considered in the program.

(3) The only experimental designs used are a 2-K factorial for first-order equation fit and a 3-K factorial for second-order equation fit.

(4) Experimental error is mainly handled by repeating the simulation test and then averaging the results. The number of repetitions are at the discretion of the user.

(5) The validation of the user friendliness of the program is accomplished by having a fellow student run the program unaided to solve a problem.

Summary

This chapter briefly discussed the general background, problem statement, research objectives, and scope pertaining to this research. The next chapter will discuss the literature and methodology pertinent to this research effort.

II. Literature And Methodology

In the past, the practical method known for handling optimization problems was the classical differential calculus. However, the classical method is impossible to use when the objective function is undefined. Therefore, an indirect method of finding the optimum using trial and error must be used. Wilde gives the name "optimum seeking procedures" to the strategies guiding search for the optimum of any function about which full knowledge is not available (9:viii).

Plan Of Attack

The only way to gain information about an unknown function is by direct measurement, in other words experimentation (5:vii). In this optimum seeking method, each experiment has two purposes, not only to attain a good response surface value, but also to give information useful for locating future experiments where desirable values of the response surface are likely to be found. Thus, throughout the search one must continually be deciding to climb or to explore. At the beginning, when nothing at all is known about the function, one must explore in some small region, usually chosen as a best guess, so that one might place the following experiments in an uphill direction. In the middle of the search, after having explored some region, one tries to climb as fast as possible, exploring only when strictly necessary to guide the successive steps. Toward the end of the search, when one is finally near the top, extensive exploration may be needed to attain any increase in elevation, the slope of the response surface often being slight near the optimum (9:64).

An analogy to this plan of attack might be like a blind man climbing to the top (highest point) of a mountain. At the bottom of the mountain, he probes around with his cane to find the steepest uphill slope. After this initial exploring, he proceeds in this uphill direction until he reaches a point where he starts to go downhill. At this point he probes around this area for a new uphill direction and proceeds uphill again. This continues until he reaches the top and can find no new uphill direction. At this time, he explores extensively around the top to find the upmost highest point. The direct search method is similar to this blind man's search in that one cannot see where one is going, but only by probing with experiments, like searching with a cane, can one get the direction to the optimum.

The three phases of the attack plan will now be discussed separately. The beginning exploratory phase to find the uphill direction will be called the gradient phase. The middle climbing phase will be called the acceleration phase. The final phase exploring the top will be called the second-order exploratory phase.

Gradient Phase

In this phase of the search, one is exploring for the uphill gradient (slope) of the response surface from the initial starting point. As mentioned earlier, if the function is known, derivatives could be taken at this point to find the gradient. Since it is unknown, another way must be determined to attain the gradient. Note--This is one of the main differences between some non-linear programs and these direct search techniques.

Wilde proposes one method of obtaining the general slope of the response surface in the neighborhood of the initial point. First, find the gradient in the x1-direction parallel to the x1 axis. To do this, one varies the x1 value slightly while holding the x2 coordinate at x20 (the initial x2 value) allowing just enough distance between x11 and x10 (the initial x1 value) to make the outcome y1 distinguishable from y0 (the initial y response value). The straight line through the points y0 and y1 lies entirely in the plane of the x20 value and is approximately tangent to the response surface at y0. The slope of this line is given by y1-y0/x11-x10. Now, a similar exploratory experiment is done, but this time varying the x2 coordinate and holding the x1 coordinate constant at x10. The straight line passing through y2 and y0 lies entirely in the plane of x10 and the slope of this line is given by y2-y0/x22-x20. The three points y0, y1, and y2 on the response surface are enough to determine the plane approximately tangent to the surface at y0. An equation of this plane would be of the form y= b0+b1x1+b2x2 (9:65-68). From the above, one can deduce the direction to proceed from the initial point is on a vector that goes y2-y0 in the x2 direction while going y1-y0 in the x1 direction. Thus, with only two experiments an approximate direction to start the climb is found.

Another exploratory search method which uses from two to four experiments is given by R. Hooke and T.A. Jeeves. This method is similar to the previous Wilde method, but is more sequential. The gradient is obtained as follows: After the initial point (x10,x20) is evaluated for y0, x10 is changed by an incremental amount, +rf, so that x11=x10+rf. If the y response is an improvement over y0, then x11 is adopted as the new coordinate in the x1 direction. If x10+rf fails to improve the response, x10 is changed by -rf and the value of the y response again checked for improvement. If the value of y is not improved by either x10 + rf, x10 is left unchanged. After the x1 direction is modified, then x20 is changed by an amount, +rf, and the above test is repeated in the x2 direction to complete one exploratory search. The successfully changed variables define a vector from the initial point for a direction to do an acceleration phase (4:142-148).

Another method to find the gradient is to use a 2-k factorial design and fit a first-order equation to the response surface using RSM procedures. What are the advantages of using a factorial design? Montgomery concludes that factorial designs are more efficient than one-factor-at-a-time experiments. Also, a factorial design is necessary when interactions may be present, to avoid misleading conclusions. Finally, factorial designs allow effects of a factor to be estimated at several levels of the other factor, yielding conclusions that are valid over a range of experimental conditions (5:192). A 2-k design would be like a box drawn around the initial point and the four corner coordinates of the box would be used to obtain response surface values (that is y values). The size of the box should be small in order to better approximate the tangent plane at the initial point, but not so small that no effective change can be seen. After obtaining the four responses, one can use RSM to fit an equation of the form, y=bo+b1x1+b2x2+b12x1x2. A good explanation of the RSM equation fitting is given by Meyers (6:43-50).

The following is how a 2-k factorial design works on the initial point (x10,x20). Let rf be the distance selected in the x1 and x2 direction for the size of the box.

The four corners would then be:

(x1n,x2p,y1h) (x1p,x2p,yhh)

(x1n,x2n,y11) (x1p,x2n,yh1)

where

x1n=x10-rf
x2n=x20-rf
x1p=x10+rf
x2p=x20+rf

The first order equation fitting these would have coefficients:

B0=(y11+y1h+yhh+yh1)/4 (1)
B1=(yhh+yh1-y1h-y11) 4 (2)
B2=(y1h+yhh-y11-yh1)/4 (3)
B12=(y11+yhh-y1h-yh1)/4 (4)

This first-order equation approximates the tangent plane at y0. B1 provides the gradient in the x1 direction and B2 the gradient in the x2 direction. However, if B12 is not zero, then there is interaction and the surface approximation is not a plane but a curved surface. Thus the B1 and B2 slopes could be misleading if the interaction is large.

The last exploratory search for a gradient to be examined is the 3-k factorial design. This design uses eight exploratory experiments and fits an equation to a second-order equation. This design provides more information about the curvature of the response surface around the initial point. The following is a description of the 3-k design. The 3-k design uses 9 points in a symmetric square pattern. In addition to the four corner points of the 2-k design, the 3-k needs four additional points.

(x1n,x2p,y1h)	(x10,x2p,ymh)	(x1p,x2p,yhh)
(x1n,x20,y1m)	(x10,x20,ymm)	(x1p,x20,yhm)
(x1n,x2n,y1l)	(x10,x2n,yml)	(x1p,x2n,yhl)

Figure 1. Nine Points of 3-K Factorial Design (6:51)

With these nine points, RSM can use this design to fit a second-order equation to this response surface. The RSM equation is of the form: $y=B0+B1x1+B11x1^2+B22x2^2+B12x1x2$. The coefficients of the equation are computed as follows:

$$B0 = (yll+ylm+ylh+yml+ymm+ymh+yhl+yhm+yhh)/9 \tag{6}$$
$$B1 = (yhl+yhm+yhh-yll-ylm-ylh)/6 \tag{7}$$
$$B2 = (ylh+ymh+yhh-yll-yml-yhl)/6 \tag{8}$$
$$B11 = (yll+ylm+ylh+yhl+yhh-2*(yml+ymm+ymh)/6 \tag{9}$$
$$B22 = (yll=yml+yhl+ylh+ymh+yhh-2(ylm=ymm=yhm)/6 \tag{10}$$
$$B12 = (yll=yhh-ylh-yhl)/4 \tag{11}$$

Similar to the 2-K design, B1 provides the gradient in the x1 direction and B2 the gradient in the x2 direction.

Thus, the literature search has provided four ways to calculate the gradient. Which of these is the most efficient for the required purpose? The unidimensional and

Hooke-Jeeves method use the fewer number of experiments; however, both calculate only one slope in each direction. If there is the least amount of error in any of the responses, it would affect the respective slopes greatly. The 2-k, using just four points, calculates two slopes in each direction and averages the result to get the gradient. Consequently, it would be more capable in dealing with any margin of error. The 3-k calculates three slopes in each direction and averages the result to get the best gradient for handling noise error, but it needs eight additional points. It needs four more points than the 2-k, but only averages 1 more slope than the 2-k. From this comparison, the 2-k design is the best. It will be used in the program to determine the gradient.

Acceleration Phase

The acceleration phase is the actual climbing up the hill. Again it is desirable to do this with as few experiments as possible. Many algorithms have been proposed on how to accomplish the ascent most effectively. Not all of the algorithms will be discussed, merely those leading up to the algorithm used in the program.

To begin with, the initial line search can be thought of as an unidimensional search along the gradient vector. The dilemma is how big of a step to take along the vector. One idea is to normalize the slopes to get a unit step along the vector. One then takes uniform unit steps along the vector. If the uniform unit step size is too small, it will take numerous steps to reach the peak of the vector. Likewise, if the step is too large, the climber might step way beyond the peak. Thus, the uniform step size is not a very efficient line search and adjusted-step line searches have been proposed as an improvement.

Robbins-Monro method was one of the first and simplest improvements over uniform step. It is based on the harmonic sequence 1,1/2,1/3,1/4, etc. times the magnitude of the dependent variable. The harmonic sequence is divergent and the sum of all its terms is infinite. Therefore, it guarantees the procedure will eventually reach the peak, no matter how far away it started (9:162-167). The problem with this line search is the exorbitant number of experiments necessary to find the peak, especially if one starts in a fairly flat region far from the optimum.

Keston has devised a procedure which accelerates the search more quickly. Instead of starting with the decreasing harmonic sequence, Keston's method starts with a uniform step then shortens the step size harmonically when the peak is crossed and the direction of search reverses. Table I compares the two methods.

Table I Unaccelerated and Accelerated Peak-Seeking Methods

steps	1	2	3	4	5	6	7	8	total
direction	+	+	+	-	-	+	-	+	
unacceler	1	1/2	1/3	1/4	1/5	1/6	1/7	1/8	1 149/280
accelerate	1	1	1	1/2	1/2	1/3	1/4	1/5	2 17/60

(9:180)

As with Robbins-Monro, the Keston method insures one of finding the peak and it achieves results more rapidly. However, it still takes numerous experiments along the vector to find the peak of the vector.

An improvement over the Keston is the golden section search. It uses a large step size to cross over the peak. Once the peak is crossed an interval exists wherein the peak is located. The golden search technique can now be used to reduce the interval of uncertainty. Golden search splits the interval with the peak into two segments such that the ratio of the whole interval to the larger segment is the same as the ratio of the larger segment to the smaller.

The golden search plan works as follows: Let the initial interval that brackets the peak be called d with endpoints of z1 and z2. Next, place 2 experiments inside this interval z3 and z4 such that z3=z1+0.38*d and z4=z1+0.62*d. If the y response of z3 is larger than that for z4, the interval of uncertainty is from z1 to z4. Otherwise, if the y-response of z4 is larger than z3, the new interval is from z3 to z2. This procedure is then continued for the new interval of uncertainty. Golden search will reduce the initial length of the interval of uncertainty by $(0.618)^{n-1}$ where n is the number of experiments used. For example, if eleven experiments were used, the new interval would be 0.008 the size of the original interval (4:42-43). In addition, Himmelblau recommends a sequential series of larger and larger steps along the vector to expedite the initial bracketing of the peak. The golden search is quite efficient compared to the previous methods and other interval uncertainty methods. Wilde provides an excellent comparison of golden section to other interval methods (9:28:29). However, there is a method of fitting a polynomial to the points that is even more efficient than golden search.

The last unidimensional line search that is examined, and the one used in this computer program, is the Davies, Swann, and Campey (DSC)-Powell Search. This method involves bracketing the peak (DSC portion) and the fitting of a quadratic equation (Powell portion) to interpolate an estimate of the peak. G.F. Coggins shows that this technique involving the fitting of a second-order polynomial through selected points was better at locating the peak to within a specified precision than the interval methods such as golden section (4:44).

The Davies, Swann, and Campey (DSC) portion is used to bracket the peak. It involves doubling the step size for each step until the peak is overshot. After the peak is overshot, the direction is reversed and previous interval used is reduced by one-half. This is used to obtain one more point. This procedure will give four equally spaced points. The two middle points y-values are compared for optimum. The point with the optimum y-value plus the two points used for fitting the quadratic since the peak should be inside this interval. See figure 2 below.

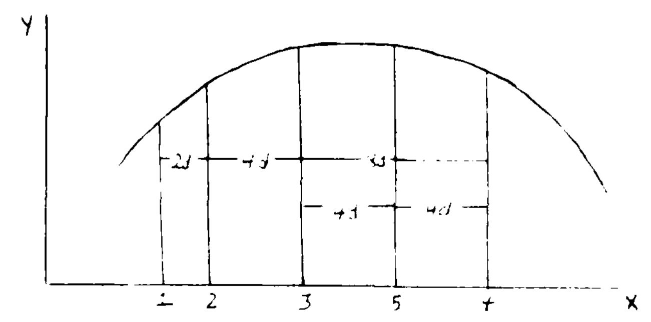


Figure 2. Davies, Swann, and Campey Technique

Powell's equation carries out a quadratic approximation using the three points. The optimum (critical point) is found by taking the first derivative of the equation. Powell's equation (4:46) is as follows:

$$x^* = -.5\frac{[(x2^2-x3^2)t1+(x3^2-x1^2)t2+(x1^2-x2^2)t3]}{(x2-x3)t1+(x3-x1)t2+(x1-x2)t3} \qquad (12)$$

The equation used in the computer program is of a slightly different form. The derivation of the computer equation is as follows:

Let p1, p2, and p3 be the three points and t1, t2, and t3 be their respective y-value. Let c be the equal distance between the points. The quadratic equation to be fitted is of the form $y=bo+b1x+b2x^2$ with derivative $dy/dx=2bx+b1=0$ implying the optimum of the equation is $x=-b1/2b2$.
Putting the three points into the equation and solving simultaneously one gets

$$b0+b1p1+b2p1^2=t1 \tag{13}$$
$$b0+b1p2+b2p2^2=t2 \tag{14}$$
$$b0+b1p3+b2p3^2=t3 \tag{15}$$

note: $p1=p1$ $p2=p1+c$ $p3=p1+2c$

by matrix notation,

$$\begin{array}{cccc} b0 & b1 & b2 & \\ \end{array}$$

$$\begin{bmatrix} 1 & p1 & p1^2 & t1 \\ 1 & p1+c & (p1+c)^2 & t2 \\ 1 & p1+2c & (p1+2c)^2 & t3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & p1 & p1^2 & t1 \\ 0 & c & 2cp1+c^2 & t2-t1 \\ 0 & 2c & 4cp1+4c^2 & t3-t1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -p1(p1+c) & (ct1-p1t2+p1t1)/c \\ 0 & 1 & 2p1+c & (t2-t1)/c \\ 0 & 0 & 2c^2 & t1-2t2+t3 \end{bmatrix}$$

note: let $d=t1-2t2+t3$ and $e=-3ct1+4ct2-ct3$

$$\begin{bmatrix} 1 & 0 & 0 & (p1^2*d+p1*e+2c^2t1)/2c^2 \\ 0 & 1 & 0 & (-2p1*d+e)/2c^2 \\ 0 & 0 & 1 & d/2c^2 \end{bmatrix}$$

$$x^* = \frac{-(-2p1*d+e)/2c^2}{2(d/2c^2)} \tag{16}$$

$$x^* = p1-.5(e/d) \tag{17}$$

This final equation is used in the computer program. It looks quite different from Powell's equation. However, if x2 and x3 are substituted by x1+c and x1+2c, respectively, then Powell's equation reduces to the one above. As mentioned earlier, the DSC-Powell method is the line search selected for the program. This selection is due to its quickness in finding the peak and its accuracy.

After finding this first point (p1) from the initial point (p0), one can repeat the gradient phase around p1 to find a new direction to proceed. The line search (DSC-Powell) is again employed to find the next point (p2). A repetition of gradient and line searches can be continued until the optimum is reached. This intuitively attractive idea of climbing the steepest path is known as the gradient method, or the method of a steepest ascent (9:107). The advantages of the steepest ascent method are: (1) It tends naturally to avoid saddlepoints,and (2) It will eventually converge for any unimodal function, even when there is appreciable experimental error (9:120-121). The steepest ascent method is one of the two algorithms of the computer program. In the program, it is called the gradient/line method.

An algorithm that accelerates faster than steepest ascent is the parallel tangent (PARTAN) method. There are several variants of the PARTAN method. The variant used is the "steepest ascent PARTAN". This method is also often referred to as "gradient PARTAN". The PARTAN method is just like steepest ascent in finding the first two points p0,and p1. After finding p1, the PARTAN method eliminates the experimental design around p1 for the next direction. The gradient to be used at p1 is the gradient perpendicular to the gradient of p0. This direction will form a plane that is parallel to the contour tangent plane of p0, where the name PARallel TANgent (abbreviated PARTAN) comes from. The perpendicular gradient can be found easily by swapping the previous slopes and reversing the sign of one of them. That is, if b1 and b2 were the slopes at p0, then now at p1 the slopes are b1=-b2 and b2=b1. A line search is then accomplished along this plane to find the peak, which is point p2. After finding p2, PARTAN eliminates another experimental design around p2 for the next direction. Instead of the 2-k factorial design, it connects a line from p0 through p2 for a new gradient direction. A line search is then accomplished starting at p2 and going along this vector direction to find p3. P3 is the optimum or very close to it. When the response surface contours are concentric ellipsoids, PARTAN will locate the optimum exactly after no more than 2k-1 unidimensional line searches (where k is the number of independent variables) (9:124). This means that point p3 (mentioned above) will be the exact optimum for a 2 independent variable deterministic problem. Thus, after one initial gradient search (4 experiments) and three line searches (about 5 experiments each), PARTAN locates the optimum. Whereas the gradient/line uses two gradient searches (4 experiments each and two line searches (about 5 experiments each) per zigzag. Therefore, one can conclude the PARTAN algorithm is indeed a more efficient method for certain quadratic problems.

Even when the contours are not precisely elliptical, PARTAN has certain ridge following properties which make it attractive especially when the ridges are straight (10:323). In addition, PARTAN will work perfectly in two dimensions for any radially similar contours since the property of parallel tangents works for these (9:144). Even for other non-ellipsoidal surfaces, PARTAN can still work. It is just that PARTAN will generally not be right at the optimum after one cycle, but this does not prevent starting over again using point p3 as the beginning of another PARTAN search.

The geometric reason PARTAN works (finds the top of the hill with so few line searches) can be simply explained using a contour plot of the response surface. See figure 3. Let a point p0 be randomly selected and a line be drawn from p0 to the center of the ellipse, p3. One will notice that this line p0p3 intersects each contour ring at the same angle. Next, the contour tangent planes, t(i)s, are drawn at the point of each of these intersections. One will observe the planes are all parallel. Also, the point of intersection with the contour ring is the optimum point along the plane line for each tangent plane. Next, the gradient vector g at p0 is perpendicular to the contour tangent plane, t0, at p0. Thus, perpendicular to all the other contour tangent planes drawn.

The PARTAN method described above would follow the darkened path in figure 3. Starting at p0, PARTAN calculates the gradient vector,g. It goes along this vector to the vector peak, p1. It then moves on a vector perpendicular to g at p1 to the vector peak, p2. It connects p0 to p2 and follows this vector to its peak, p3. This described method will be called the PARTAN/line method for the rest of this paper and in the computer program.

There is still another improvement to the search method. Faix proposes an efficient improvement upon the PARTAN/line method. Faix states,

> The method only works exactly for perfect quadratic response surfaces with no noise. However, it will be shown to be relatively robust against many types of imperfection, and thus a good methodology choice [1:180].

This improvement, to be called the PARTAN/FAIX method, eliminates the line search from p2 to p3. The PARTAN/FAIX

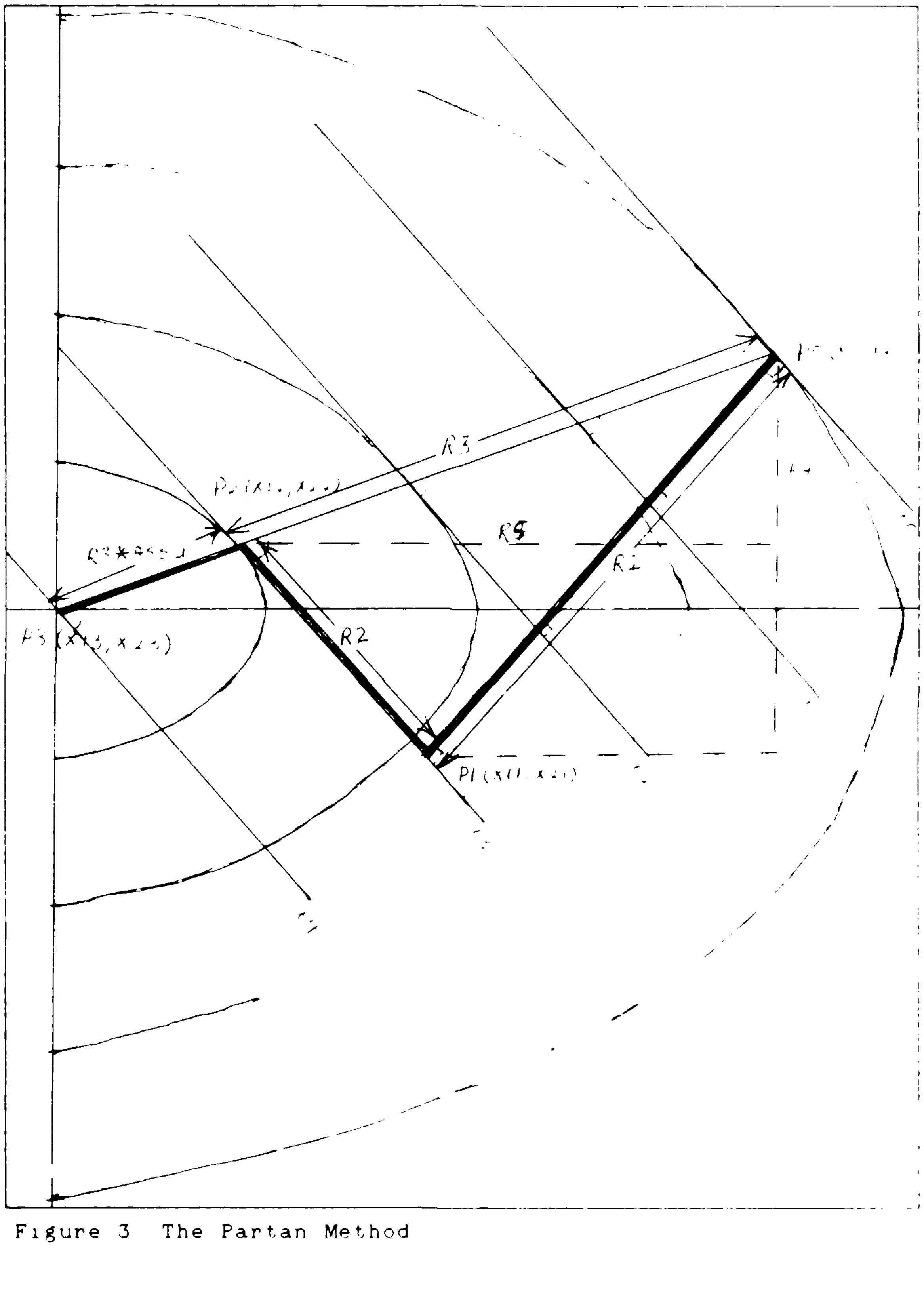


Figure 3 The Partan Method

method calculates the distance from p2 to p3 by using the results from points p0, p1, and p2 to find the eccentricity of the ellipse. The eccentricity, e, measures the stretch of the ellipse. The eccentricity of an ellipse is equal to c/a in figure 4 and ranges from 0 to almost 1.

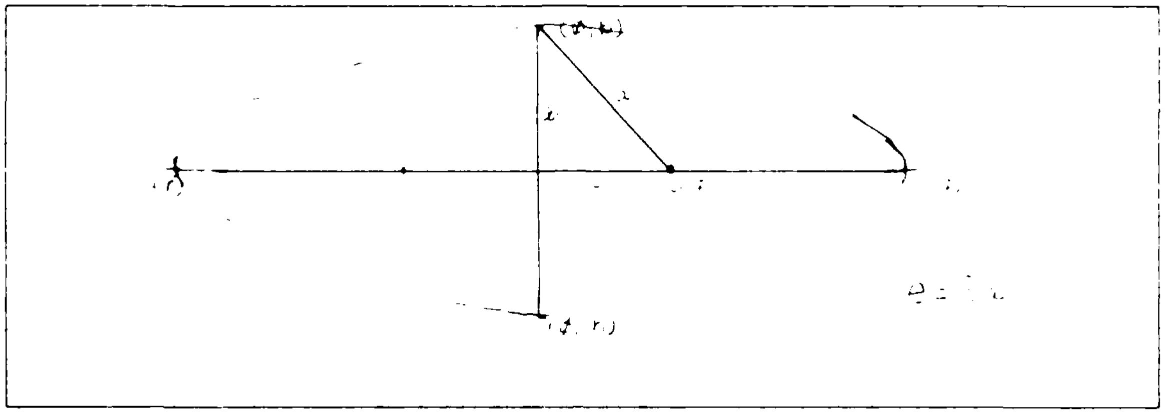


Figure 4. Eccentricity of an Ellipse

The eccentricity is zero for a circle and approaches one as the major diameter increases in ratio to the minor diameter (10:379-399). For a quadratic equation, $y=bo+b1x1+b2x2+b11x1^2+b22x2^2$, one can relate the coefficients, b11 and b22, to e. The square root of (b22/b11) is equal to the a/b in figure 4. Therefore, if b22 is greater than b11, then e is equal to the square root of (1-b11/b22) and if b11 is greater than b22, then e is equal to the square root of (b22/b11-1). In agreement with Faix's notation, b22/b11 will be called the variable c (1:186).

The variable c can be found geometrically using the points p0, p1, and p2 of a PARTAN search. Using figure 3,

$$c=(1+mo*r)/(mo*r-(mo)**2) \tag{18}$$

where

mo=	r4/r5, the known slope of line r3 between p0 an p3
r=	r1/r2, the ratio of lines r1 and r2
r1=	the distance between p0 and p1
r2=	the distance between p1 and p2
r3=	the distance between p0 and p2
r4=	the distance x22-x20
r5=	the distance x12-x10 (1:182)

Using the variables c, mo, and r3, Faix derives the length of the acceleration step between p2 and p3 in multiples of r3. The length between p2 and p3 equal r3*assu where (1:182)

$$assu=[(mo)^{2}*(c-1)^{2}*c]/[1+(c*mo)^{2}]^{2} \tag{19}$$

There is an equally efficient method to the PARTAN/FAIX method. One may notice that instead of using the parallel plane, t3, any of the other parallel planes would have worked for PARTAN. Thus, instead of doing a line search from p0 to find p1, choose any distance to place p1 from p0. A unit step of 1 is offered as an option in the computer program. Then do a line search perpendicular to find p2. Connect p0 and p2 and do a line search in this direction to find p3. This would entail just two line searches comparable to the PARTAN/FAIX method.

Second-Order Exploratory Phase

Both the beginning and the end of a search involve local exploration. The beginning being a simple linear study near an arbitrary point to get to the gradient direction. At the end of the search, a nonlinear exploration in the vicinity of the optimum is accomplished to insure the optimum was found (9:75). This end exploration may actually find a point nearby that is better then the optimum found by the algorithm. This could be caused by error in the simulation or testing process. In addition, this final exploratory phase will show the behavior of the response surface near the optimum.

A 3-k factorial design is used to provide the second-order equation fit. This design was discussed under the gradient phase. The difference now is once the coefficients are found, they are fitted into a derived equation for the critical point.

$$x1f=x10+[(-b2*b12)+(2*b11*b1)]/[(b12*b12)-(4*b11*b22)] \qquad (20)$$

$$x2f=x20+[(-b1*b12)+(2*b11*b2)]/[(b12*b12)-(4*b11*b22)] \qquad (21)$$

If $(4*b11*b22)$ is less than $(b12*b12)$, then this point is a saddle point. Otherwise, if b11 is less than 0 and b22 is less than 0, then the point is a maximum. If b11 and b22 are positive, then the point is minimum. Thus, the coefficients b11 and b22 describe the shape of the response surface around the optimum.

Summary

This chapter reviewed the literature and methodology that lead up to the writing of the computer program. The next chapter describes the actual computer program and how it works.

III. Program Description

This chapter describes the flow of the computer program. The complete program is contained in Appendix A. The program is written in FORTRAN on a VAX computer but could be transferred to a microcomputer for further use.

Before running the program, the subroutine SIM must be modified in two ways:

(1) Line 18 of the subroutine must reflect whether the problem is a minimization or maximization. This is done by removing the letter c in the first column of line 18 for a minimization and ensuring the letter c is in place for a maximization. The letter c comments out line 18 for a maximization problem.

(2) The problem simulation must be loaded into the SIM subroutine starting at line 14. If the simulation or experimentation is to be run externally of the program, column 1 in line 14 gets a letter c added and column 1 in lines 15,16, and 17 get the letter c removed.

Main Program

The main program is called SEARCH. A flow diagram of SEARCH is shown in figure 5. The program has numerous interactive options for the user to allow the user the freedom to work a variety of problems. However, the program is mainly designed to efficiently optimize quadratic problems.

SEARCH consists of repetitions of asking the user to make a choice and then calling the subsequent subroutines and showing the outcome for that choice. This allows the program to step along from point to point toward the optimum.

The program begins by asking the user for the point 0 x1 and x2 starting coordinates. It then calls the subroutine SIM, which gives the y response for these inputs and the main program writes these values to the screen and output file. At this point, it automatically calls the subroutine TWOK. TWOK does a 2-k factorial design and RSM fit to find the gradient directions. If the linear equation is of a flat surface, thus having no gradients, the main program will end for there is no direction to climb at this point. Otherwise, the main program will write to the screen and the output file, the normalized gradient directions. This gradient direction is the best direction for climbing.

Next the main program will prompt the user for how far to travel in this gradient direction. The two options are: one unit step or to the peak in that direction. The first option should be used only if the response surface is ellipsoidal. The second choice might be used with PARTAN or the steepest ascent method. With the choice made, the program will calculate point 1 and write the location of point 1 to the screen and output file.

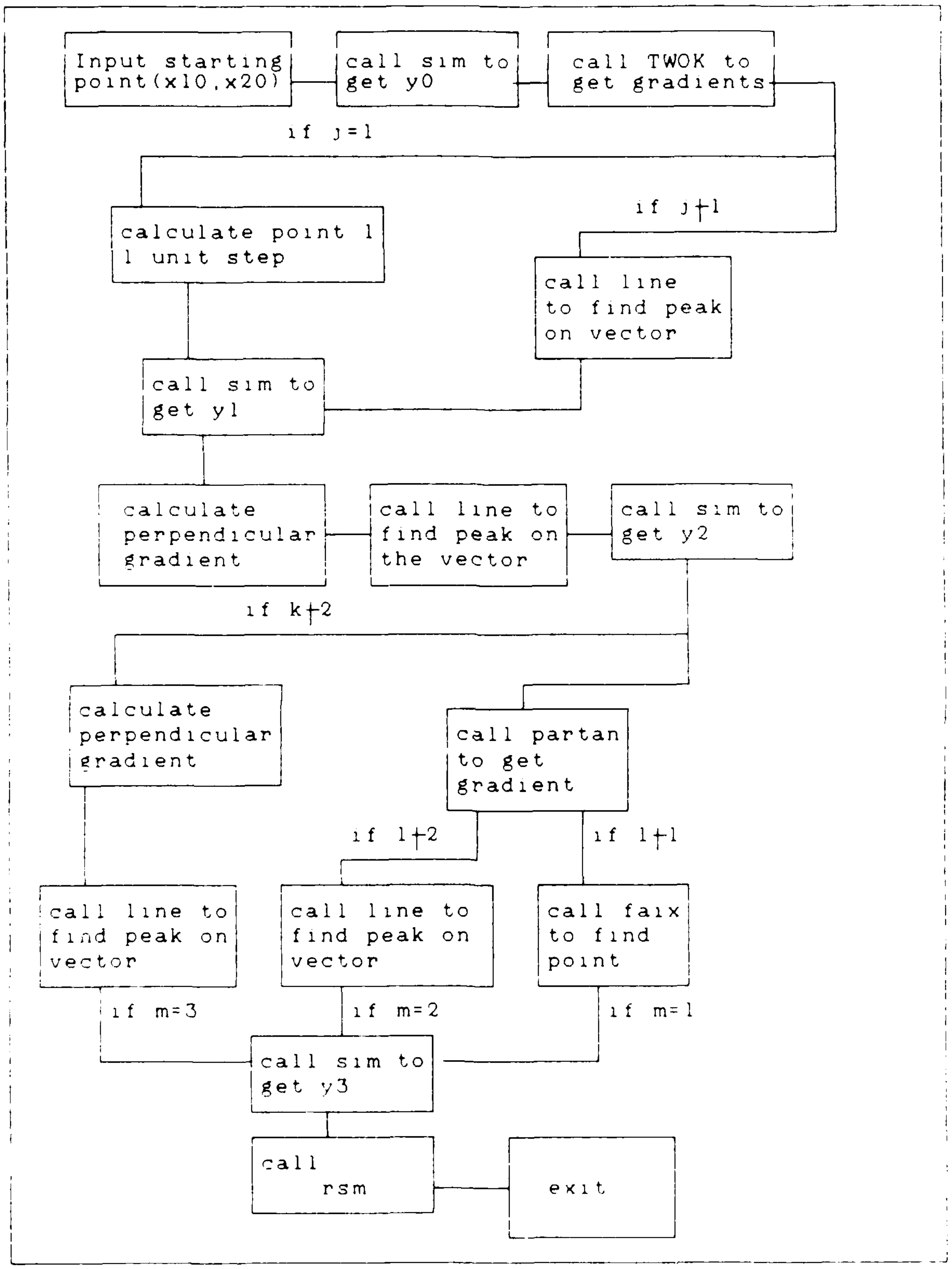
Proceeding from point 1, the main program calculates the gradient perpendicular to the last gradient direction and writes it to the screen and output file. It then calls subroutine LINE to find the peak along that gradient line. After calling SIM, it writes the calculated point 2 to the screen and output file.

At point 2, the user decides to use the PARTAN method or continue the gradient/line method. If option 1 (gradient /line) is selected, then the perpendicular gradient is calculated and written. The program calls subroutine LINE to find the peak in this direction. This is followed by the subroutine SIM. The grad/line point 3 is then written to the screen and output file. If option 2 (PARTAN) is selected, then the subroutine PARTAN is called to calculate the gradients. These two gradients are written to the screen. With the PARTAN directions, the program offers the user the option of doing a line search or a FAIX jump to the next point. If a one unit step was selected at point 0, then a line search must be selected. Otherwise, the second option (FAIX method) is the most efficient and, if selected, the main program calls the subroutine FAIX to compute the PARTAN/FAIX point 3.

Next, the main program offers the user to choose which of the previous three options he wants to use for point 3.

This is included in case more than one option was selected. Point 3 is then written to the screen and the output file.

Finally, the main program asks the user if he wants to exit at this point, repeat the whole process again using point 3 as the new initial point, or to do a 3-k factorial design and RSM to better locate the final point. If the user is confident the surface is ellipsoidal and there was little error in the input values, then one should be at the optimum and exiting is the correct choice. If the user is sure the optimum has not yet been reached, maybe due to the complexity of the surface, then repeating the process again would produce the better answer. If the user feels close to the optimum, but point 3 is slightly off, then 3-k design with RSM will help to find the final optimum.


Input starting point(x10,x20)
call sim to get y0
call TWOK to get gradients
if j=1
if j≠1
calculate point 1 1 unit step
call line to find peak on vector
call sim to get y1
calculate perpendicular gradient
call line to find peak on the vector
call sim to get y2
if k≠2
calculate perpendicular gradient
call partan to get gradient
if l≠2
if l≠1
call line to find peak on vector
call line to find peak on vector
call faix to find point
if m=3
if m=2
if m=1
call sim to get y3
call rsm
exit
Figure 5. Flow Diagram of Main Program

TWOK Subroutine

This subroutine fits a 2-k factorial design around a point and uses RSM to fix an equation to the four points. The coefficients of the equation are used for the gradients. Figure 6 is the flow diagram for the TWOK subroutine. The subroutine begins by stating the radius of the 2-k factorial design and asking the user if he would like to change the radius size. After the radius size is determined, the coordinates of the four corner points are calculated. With these coordinates, the subroutine SIM is called and the response values found. Next, the maximum y-value of the four points is called the variable, m. This variable is compared to the initial point response, y0. If y0 is larger than m, then there appears to be no uphill direction from the initial point. Therefore, the subroutine RSM is called for an exploratory search of the optimum within this area. Otherwise, the four y-responses are used to calculate the coefficients of the first-order equation. Of these four coefficients b0, b1, b2, and b12, b1 and b2 are used as the x1 slope and x2 slope, respectively. If both of these values are zero, then there is no slope in this area and the program will print "Try a new starting point." and end. Before b1 and b2 are passed back to the main program, the subroutine normalizes their value. The control then returns to the main program.

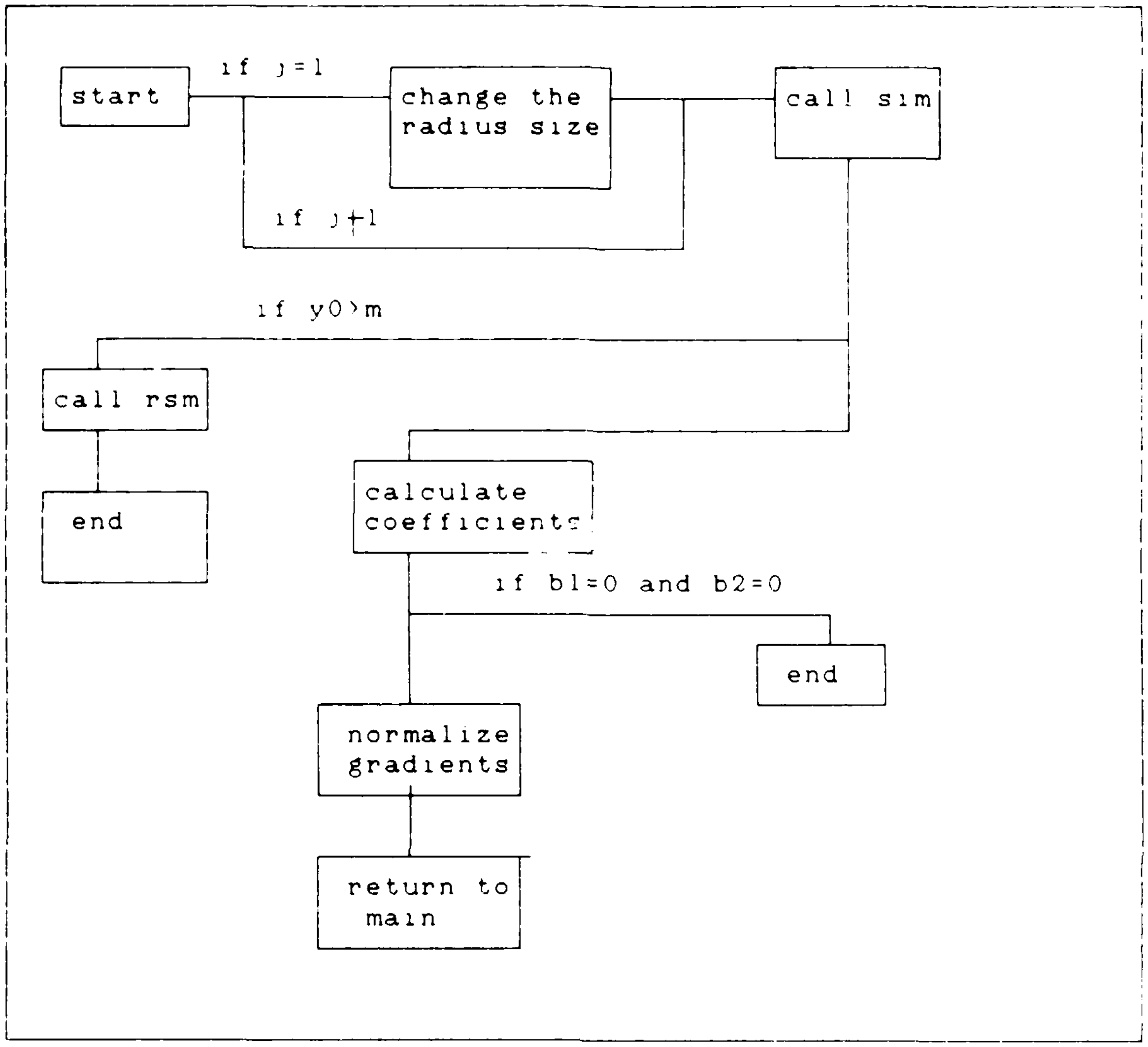


Figure 6. Flow Diagram of TWOK Subroutine

LINE Subroutine

This subroutine finds the peak in a vector direction using the DSC-Powell algorithm. Figure 7 is the flow diagram for the LINE subroutine. The subroutine first calculates a point that is 2 units in the gradient direction from the starting point. After calling SIM to get the y-response, it checks to see if the response was an increase over the initial response. If it was not an increase, the subroutine reverses the gradient direction and calculates a point 2 units in the other direction. It again calls SIM to get the y-response. If this too was not an increase, the program curve fits these three points to find an optimum. If either direction had been an increase response, the subroutine would double the step size and calculate the next point. It would continue doubling the step size until it has either gone 10 steps(to prevent a continuous loop) or got a response that was a decrease from the previous step. Once it gets a y-response that is a decrease, it cuts the step size in half and reverses the vector direction. This gives four equally spaced points. The subroutine compares the 2 middle responses. It uses the point with the larger response and the points on both sides of it to curve fit an equation. The first derivative of this equation is used to find the optimum point along the vector. This optimum point is then returned to the main program.

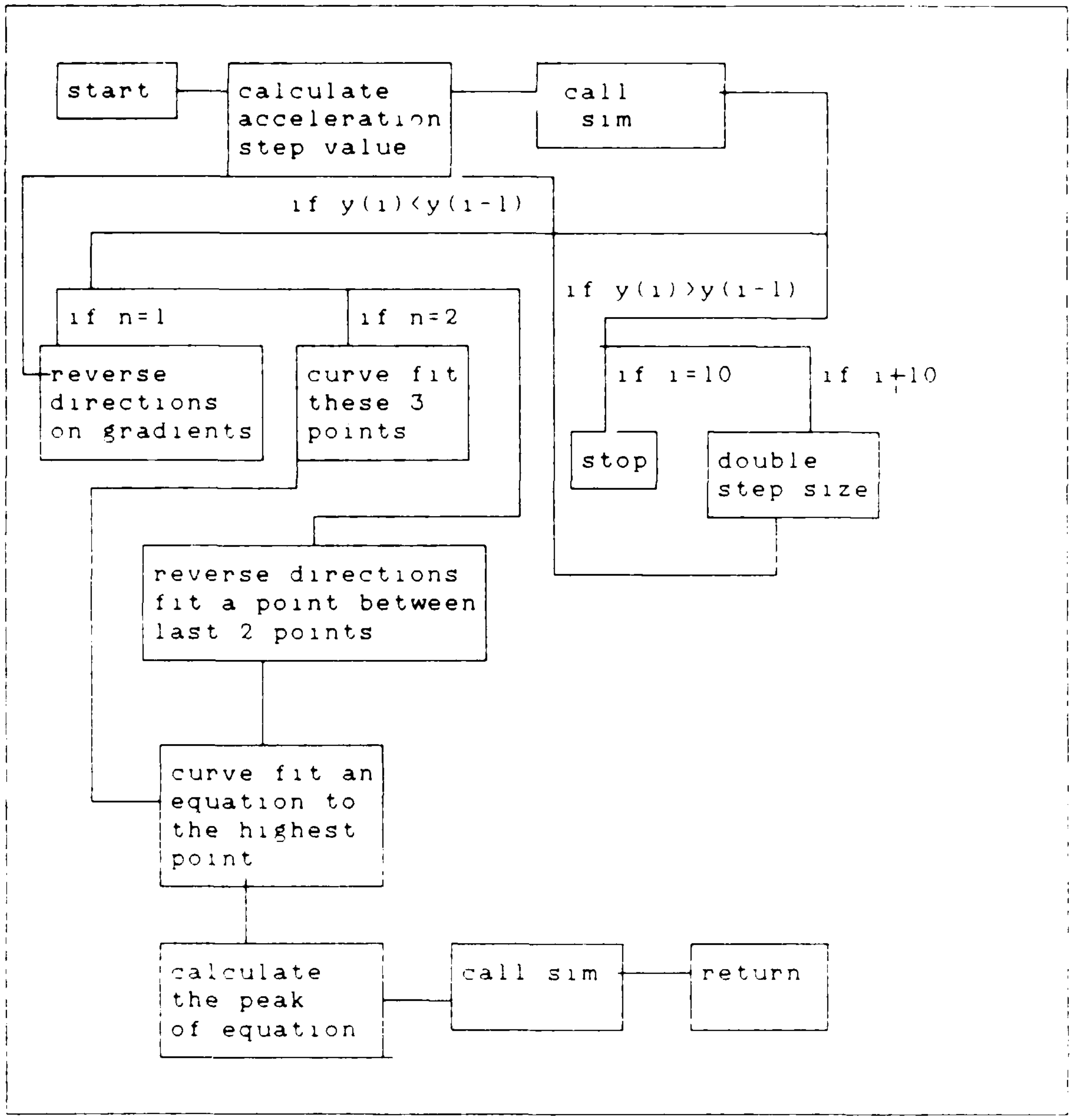

start
calculate acceleration step value
call sim
if y(i)<y(i-1)
if y(i)>y(i-1)
if n=1
if n=2
reverse directions on gradients
curve fit these 3 points
if i=10
if i≠10
stop
double step size
reverse directions fit a point between last 2 points
curve fit an equation to the highest point
calculate the peak of equation
call sim
return

Figure 7. Flow Diagram of LINE Subroutine

SIM Subroutine

This subroutine ties the simulation model or user experimental values with the program. See figure 8 for a flow diagram for the SIM subroutine. The subroutine begins by asking the user how many repetitions of the problem is to be accomplished at these values. The default is 1 simulation. It then loops through the simulation or user input the required number of repetitions. The subroutine then averages the responses to get one value to pass back to the program. Also, by enabling line 18 the program can run minimization problems by doing a negative maximization.

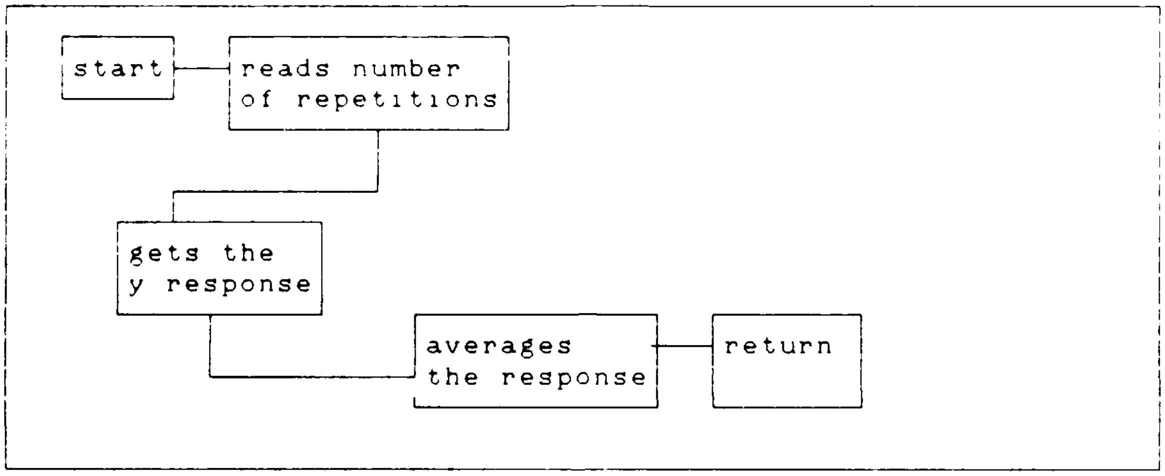


Figure 8. Flow Diagram of SIM Subroutine

PARTAN Subroutine

This short subroutine takes the coordinates of point 0 and point 2 and calculates the slope between these two points. This slope is then normalized and passed to the main program to be used as the next gradient direction.

FAIX Subroutine

This subroutine takes the coordinates of points 0, 1, and 2 and calculates the distances between these points. It also calculates the slope between point 0 and point 2. It then computes the ratio of the distance between p0 and p1 over the distance between p1 and p2. With these values, the subroutine determines values for the variables c and assu. Finally, it uses assu and the distance between p0 and p2 to estimate the location of the p3 coordinates. After calling SIM subroutine to get the y-response for point 3, it returns to the main program.

RSM Subroutine

This subroutine fits a 3-k factorial design around a point and uses RSM to fit a second-order equation to the nine points. The first derivative of the fitted equation is used to find the critical point in this area. The second derivative test is then used to determine whether the critical point is a maximum, minimum, or a saddle point.

Summary

This chapter described the procedures of the SEARCH program. It looked at the contents of each subroutine andthe flow of the main program. The next chapter discusses how well the program works.

IV. Validation of Program

A validation of the program will be accomplished by taking a known quadratic equation and comparing the SEARCH program output with the known mathematical values. See figure 9 for the output file. The equation is $y = 10-5(x1+1)^2-15(x2-2)^2$. An initial point of (6,9) will be used to start the program.

The first check is of the gradient around the initial point. The SEARCH program obtains gradients of (-0.3162278,-0.9486833). The first derivatives of the equation are $(-10(x+1),-30(x-2))$ and at the initial point (6,9)would yield (-70,-210) as the gradients. If (-70,-210) is normalized, one gets exactly the same values as the SEARCH program obtains. Thus, the TWOK subroutine does obtain accurate gradients.

The next check is of the line search to point 1. The SEARCH program used four steps to get to point 1. It used three steps before it passed the vector peak and one reverse step to evenly space the points. After curve fitting, the program obtained point 1 as (3.5,1.5). This point 1 is 7.9 unit gradient steps from point 0 with a y-value of -95.0003. To check the accuracy of this line search, one can check 7.8 and 8.0 unit gradient steps. This gives y-values of -95.156 and -95.1245, respectively. Thus, the line search was quite accurate in selecting the peak value along the gradient vector.

After point 1, the program computes gradients that are perpendicular to the first set of gradients. One can see by inspection that the new gradients (-0.949,0.316) are perpendicular. Thus the calculation is correct.

Another line search is accomplished after just three steps to find point 2. The SEARCH program after three steps finds point 2 as (-0.25,2.75) with a y-value of -1.250020. Using 3.9 and 4.0 gradient steps to check the accuracy, one gets y-values of -1.2657 and -1.26336, respectively. Thus the line search is very accurate again.

Next, the program calculates the PARTAN gradient as (-0.707,-707). This is the slope between point 2 and point 0 (2.75-9,-.25-6) and are the same values once normalized.

Finally, the FAIX subroutine calculates point 3 as (-1,2) with a y-response of 10. This can be checked by the format of the equation as the actual optimum. The RSM subroutine is also ran, but shows it is unable to improve the optimum.

This demonstration by example has shown the accuracy of the subroutines that make up the program. The output from other problems are contained in Appendix B.

The efficiency of the SEARCH program is evident by the few simulation runs required. The above problem needed a

```
Problem: y=10-5(x1+1)**2 -15(x2-2)**2

 point 0=   6.000000        9.000000       -970.0000
   5.500000        8.500000        -835.0000
   5.500000        9.500000        -1045.000
   6.500000        8.500000        -905.0000
   6.500000        9.500000        -1115.000
 b0,b1,b2,b12 are
  -975.0000      -35.00000      -105.0000      0.0000000E+00
 slope in x1 direction is -0.3162278
 slope in x2 direction is -0.9486833
   5.367545        7.102633        -583.2812
   4.102633        3.307900        -145.8434
   1.572811       -4.281567        -614.9680
   2.837723      -0.4868333        -156.4057
 point 1 =    3.500000        1.500000        -95.00003
 the new slope for x1 is  0.9486833
 the new slope for x2 is -0.3162278
   5.397367       0.8675440        -213.8684
   1.602634        2.132455        -24.13168
  -2.192100        3.397366        -26.39499
 -0.2947329        2.764911        -1.263333
 point 2 is -0.2500001        2.750000        -1.249993
 partan slopes are -0.7071068      -0.7071068
 r1,r2,r3,r,mo are
   7.905694   3.952848   8.838835   2.000000   1.000000
 c=   3.000001
 assu=  0.1200000
 faix point 3 is  -1.000000    2.000000    10.00000
 point 3 is  -1.000000        2.000000        10.00000
  -1.500000        1.500000        4.999996
  -1.500000        2.000000        8.749999
  -1.500000        2.500000        5.000003
  -1.000000        1.500000        6.249997
  -1.000000        2.000000        10.00000
  -1.000000        2.500000        6.250004
 -0.5000001        1.500000        4.999998
 -0.5000001        2.000000        8.750001
 -0.5000001        2.500000        5.000005
 b0,b1,b2,b11,b22,b12 =
6.666667  1.1126200E-06  3.6557515E-06  -1.250000  -3.750000
  1.1920929E-07
 the final point is  -1.000000       2.000000       10.00000
 (x1f,x2f) is a maximum point.
```

Figure 9 Output from Sample Problem

mere twelve simulations to find the optimum plus eight additional to run the RSM accuracy check at the end.

A comparison of this program to other similar programs is not realistic Most other programs are written to solve complex, nonlinear equations. Reklaitis compares several of these methods and algorithms, and was unable to determine a superior method (7:60,120).

Summary

This chapter used an example problem as a validation of the program. It showed the accuracy of the subroutines and efficiency of the program. The next chapter recommends further enhancements.

V. Conclusion and Recommendations

The overall objective of this research was to develop an interactive, user-friendly computer package that allows one to locate the optimum response of an unknown objective function in a minimum number of experimental trials

Subobjectives were:

(1) Show that the techniques chosen were the most efficient using the least number of trials as the measure of effectiveness.

(2) Verify that the program can find the optimal response to a sample problem.

This research effort accomplished these objectives. The program is user-friendly and solves the optimizationproblem in a very efficient number of trials. It, in addition, provides the flexibility to solve even more complex problems than just quadratic surfaces.

Recommendations for further enhancements would be to expand the number of independent variables that the program can handle. This would increase the base of problems the program can solve. Another enhancement would be to enable the program to incorporate constraints. This would broaden its adaption to real world problems. Finally, one last user-friendly enhancement would be to add its own graphic display of the response surface.

Appendix A: Program Listing

```
        program search

        integer g,i,j,k,l,m

        real x10,x20,y0,b1,b2,r1
        real x11,x21,y1,x12,x22,y2,x13,x23,y3
        real f13,f23,f3,l13,l23,l3
        real x1f,x2f,yf

        open(7,file='output',status='unknown')

c------this portion interactively gets the starting point.
        print *, 'Input the starting point.  This should'
        print *, 'be your best guess of the optimum.'
        print *, 'Enter the x1 coordinate.'
        read *,x10
        print *, 'Enter the x2 coordinate.'
        read *,x20
        call SIM (x10,x20,y0)
        print *, 'The value for y0 is',y0,'.'
200     write(7,*),'point 0=',x10,x20,y0


110     print *, 'Around this initial point(x10,x20) a 2K
factorial'
        print *, 'design is accomplished to get the '
        print *, 'gradient (direction of ascent).'
c------this subroutine calculates the first gradient.
        call TWOK (x10,x20,y0,b1,b2)
c------this if statement is for flat surface
         if ((b1.eq.0).and.(b2.eq.0)) then
          go to 1000
         end if
        print *, 'slope in x1 direction is',b1
        print *, 'slope in x2 direction is',b2
        write(7,*),'slope in x1 direction is',b1
        write(7,*),'slope in x2 direction is',b2

        print *, 'Enter 1) to go 1 unit step'
        print *, 'Enter 2) to do a line search'
        read *, j
        if (j .eq. 1) then
         x11=x10+b1
         x21=x20+b2
         call SIM (x11,x21,y1)
         go to 300
        end if

c------this subroutine does a line search for the next
point.
```

```
       print *, 'Using the gradients, a line search will be'
       print *, 'accomplished to find the peak in this
direction'

       call LINE (x10,x20,y0,b1,b2,x11,x21,y1,i)
c------this if statement is for lines that never peak
        if (i.eq.10) then
         go to 1000
        end if
300    print *, 'The x11 coordinate is',x11
       print *, 'The x21 coordinate is',x21
       print *, 'The value for y1 is',y1,'.'
       write(7,*),'point 1 =',x11,x21,y1

       print *, 'Do you want to do a 2-k design to '
       print *, 'get the next gradient or use the
perpendicular'
       print *, 'to the line search.'
       print *, '1) perpendicular'
       print *, '2) 2-k factorial design'
       read *,g
       if(g .eq. 1)then
       r1=b1
       b1=-b2
       b2=r1
       go to 25
       end if

c------this step gets a gradient at this location.
       print *, 'Another 2-k factorial will be done'
       print *, 'to find gradients from this point.'

       call TWOK (x11,x21,y1,b1,b2)
        if ((b1.eq.0).and.(b2.eq.0)) then
         go to 1000
        end if
25     print *, 'slope in x1 direction is',b1
       print *, 'slope in x2 direction is',b2
       write(7,*),'the new slope for x1 is',b1
       write(7,*),'the new slope for x2 is',b2

c------this step does a line search for the next point.
       print *, 'A line search will be done using these'
       print *, 'gradients from point 1.'
       call LINE (x11,x21,y1,b1,b2,x12,x22,y2,i)
        if (i.eq.10) then
         go to 1000
        end if
       print *, 'The x12 coordinate is',x12
       print *, 'The x22 coordinate is',x22
       print *, 'The value for y2 is',y2,'.'
       write(7,*),'point 2 is',x12,x22,y2
```

```
c------the next step lets one choose between another
gradient/line
c       search or the shorter partan.
        print *, 'Do you want to do another gradient/line
search or'
        print *, 'the partan method or both?'
        print *, 'The partan connects point 0 and 2 for the
gradient.'
        print *, '    1) yes - gradient/line'
        print *, '    2) yes - partan'
        print *, '    3) yes - compare both'
        print *, 'enter 1,2,or3.'
        read *,k
        if (k.ne.2) then
           call TWOK (x1 ,x22,y2,b1,b2)
           print *, 'twok b1=',b1
           print *, 'twok b2=',b2
           write(7,*),'twok slopes are',b1,b2

           call LINE (x12,x22,y2,b1,b2,x13,x23,y3,i)
           print *, 'gradient/line new point location is'
           print *, x13,x23,y3
          write(7,*),'grad/line point 3 is',x13,x23,y3

        end if
        if (k.ne.1) then
           call PARTAN (x10,x20,x12,x22,b1,b2)
           print *, 'partan b1 =',b1
           print *, 'partan b2 =',b2
           write(7,*),'partan slopes are',b1,b2

         print *, 'do you want to do a line search or the
faix method?'
           print *, ' 1) yes - line search'
           print *, ' 2) yes - faix method'
           print *, ' 3) yes - compare both'
           print *, 'enter 1,2,or3.'
           read *,l
            if (l.ne.1)then
            call FAIX
(x10,x20,x11,x21,x12,x22,b1,b2,f13,f23,f3)
            print *, 'the FAIX method calculated'
            print *, f13,f23,f3
            write(7,*),'faix point 3 is',f13,f23,f3

            end if
            if (l.ne.2) then
            call LINE (x12,x22,y2,b1,b2,l13,l23,l3,i)
            print *, 'the line search found'
            print *, l13,l23,l3
            write(7,*),'partan/line point 3 is',l13,l23,l3
            end if
          end if

            print *, 'which do you want to use?'

            print *, 'enter 1 to use partan/FAIX values (if
used)'
            print *, f13,f23,f3

            print *, 'or 2 for partan/line values (if used)'
            print *, l13,l23,l3

            print *, 'or 3 to use gradient/line values (if
used)'
            print *, x13,x23,y3

            read *,m
             if (m.eq.1) then
              x13=f13
              x23=f23
              y3 =f3
             else if (m.eq.2) then
              x13=l13
              x23=l23
              y3 =l3
             end if
       print *, 'The x13 coordinate is',x13
       print *, 'The x23 coordinate is',x23
       print *, 'The value for y3 is',y3,'.'
       write(7,*), 'point 3 is',x13,x23,y3
       print *, 'which do you want to do?'
       print *, '1) quit/exit'
       print *, '2) repeat process using point 3 as initial
point'
       print *, '3) 3-k factorial design and RSM'
       read *,h
       if (h .eq. 1)then
         go to 1000
       else if (h .eq. 2) then
         x10=x13
         x20=x23
         y0=y3
         go to 200
       end if

       call RSM (x13,x23,y3,x1f,x2f,yf)
1000   print *, 'the end'

       end

       subroutine TWOK(x10,x20,y0,b1,b2)
```

```
integer j
 real x10,x20,y0,rf,x1n,x2n,x1p,x2p
 real *y11,*y1h,*yhh,*yh1,b1,b2,b12,b0,norm,m
 real x1f,x2f,yf
 rf=0.5

 print *, 'Entering 2-k factorial.'

 print *, 'The radius of the factorial design is',rf
 print *, 'Would you like to change this radius ?'
 print *, ' 1) yes   2) no'
 read *, j
 if (j.eq.1) then
  print *, 'Enter the new radius .'
  read *,rf
 end if

 x1n=x10-rf
 x2n=x20-rf
 x1p=x10+rf
 x2p=x20+rf

 call SIM(x1n,x2n,*y11)
 print *, '*y11 =',*y11
 write(7,*),x1n,x2n,*y11

 call SIM(x1n,x2p,*y1h)
 print *, '*y1h=',*y1h
 write(7,*),x1n,x2p,*y1h

 call SIM(x1p,x2n,*yh1)
 print *, '*yh1 =',*yh1
 write(7,*),x1p,x2n,*yh1

 call SIM(x1p,x2p,*yhh)
 print *, '*yhh =',*yhh
 write(7,*),x1p,x2p,*yhh

  m= amax1(*y11,*y1h,*yh1,*yhh)
  print *, 'm=',m

  if (y0.gt.m) then
   print *, 'yo is larger'
   call RSM (x10,x20,y0,x1f,x2f,yf)
   b1=0
   b2=0
   go to 70
  end if

  b0=(*y11+*y1h+*yh1+*yhh)/4
  b1=(-*y11-*y1h+*yh1+*yhh)/4
```

```
          b2=(-*yll+*ylh-*yhl+*yhh)/4
          b12=(*yll-*ylh-*yhl+*yhh)/4
          print *, 'b0=',b0
          print *, 'bl=',bl
          print *, 'b2=',b2
          print *, 'b12=',b12
          write(7,*),'b0,bl,b2,b12 are'
          write(7,*),b0,bl,b2,b12

          print *, 'note interaction of b12 '

          if((bl.eq.0).and.(b2.eq.0)) then
            if (b0 .eq. y0) then
             print *, 'the surface is flat in this area'
             print *, 'y=',b0
            else if (y0 .gt. b0) then
             print *, 'max point in this area is ',y0
            else if (y0 .lt. b0) then
             print *, y0, 'is a minimum point in this area'
            end if
          print *, 'try a new starting point'
          go to 70
          end if

          norm=((bl**2)+(b2**2))**0.5
          bl=bl/norm
          b2=b2/norm

70       continue
         return
         end



         subroutine SIM(xl,x2,y)
          real xl,x2,y,w(10),u,v
          integer rep,i
           rep=1

c          print *, 'Enter the number of repetitions of the
simulation'
c          print *, 'wanted at this point to reduce
experimental error.'
c          read *,rep
          y=0
          do 20 i=1,rep
          u=xl
          v=x2

c------this is the place to insert the simulation.
          w(i)=100*(v-u**2)**2+1*(1-u)**2
```

```
c------the next line changes the problem to a minimization
c------remove the c in column 1 of line 18 to minimize
c------insure the c is in column 1 of line 18 to maximize
 18          w(i)=-w(i)

            y=y+w(i)
20          continue
            y=y/rep
            print *,x1,x2,y
            return
            end

            subroutine LINE(x10,x20,y0,b1,b2,z1,z2,yz,i)

            integer i,n
            real x1(0:10),x2(0:10),y(0:10),b1,b2
            real x10,x20,y0,t1,t2,t3,c,d,e,f,z1,z2,yz
            real s1,s2,s3
            n=1

            print *, 'Entering line search.'
            i=0
            x1(i)=x10
            x2(i)=x20
            y(i)=y0

10          i=i+1
5           x1(i)=x1(i-1)+((2**i)*b1)
            x2(i)=x2(i-1)+((2**i)*b2)
            s1=x1(i)
            s2=x2(i)
            call SIM (s1,s2,s3)
            y(i)=s3

            print *, x1(i)
            print *, x2(i)
            print *, y(i)
            write(7,*) x1(i),x2(i),y(i)

            if(y(1).lt.y0)then
            print *, 'y1 lt y0'

              if(n.eq.1)then
                b1=-b1
                b2=-b2
                t3=y(1)
                n=2
               print *, 'b1,b2,t3,n are'
               print *, b1,b2,t3,n

                go to 5
              else if(n.eq.2)then
```

```
        t1=y(1)
        t2=y0
        c=-2.0
      print *, 't1,t2,c are'
      print *, t1,t2,c

        go to 15
       end if
    end if

    if( y(i) .gt. y(i-1)) then
     if (i .eq. 10) then
     print *, 'the surface has increased for 10 steps'
     print *, 'in this direction.  It appears to go to'
     print *, 'infinity for a optimal point.'
     print *, 'start with a new point.'
     go to 80
     else
     go to 10
     end if
    else
      i=i+1
      x1(i)=x1(i-1) - (b1* 2**(i-2))
      x2(i)=x2(i-1) - (b2* 2**(i-2))
      call SIM (x1(i),x2(i),y(i))
      print *, x1(i)
      print *, x2(i)
      print *, y(i)
      write(7,*) x1(i),x2(i),y(i)
    end if
       if (y(i) .ge. y(i-2)) then
        t1=y(i-2)
        t2=y(i)
        t3=y(i-1)
       else
        t1=y(i-3)
        t2=y(i-2)
        t3=y(i)
       end if

      print *,t1,t2,t3

      c=2**(i-2)
15    print *, 'hello'
       d=t1-(2*t2)+t3
        if(d.eq.0)then
          d=.0000001
        end if
       e=(-3.0*c*b1*t1)+(4.0*c*b1*t2)-(c*b1*t3)
       f=(-3.0*c*b2*t1)+(4.0*c*b2*t2)-(c*b2*t3)
      print *,c,d,e,f
```

```
              if (tl .eq. y(i)) then
               z1= x1(i) - 0.5*e/d
               z2= x2(i) - 0.5*f/d
              else if (tl .eq. y(i-2)) then
               z1=x1(i-2) -0.5*e/d
               z2=x2(i-2) -0.5*f/d
              else if (tl .eq. y(i-3)) then
               z1= x1(i-3) - 0.5*e/d
               z2= x2(i-3) - 0.5*f/d
              end if
            call SIM (z1,z2,yz)

60          print *,z1,z2,yz

80          continue
            return
            end

          SUBROUTINE PARTAN (x10,x20,x12,x22,b1,b2)

           real x10,x20,x12,x22,b1,b2,norm

           print *, 'Entering PARTAN.'

           b1= x12 - x10
           b2= x22 - x20
           norm = ((b1**2) + (b2**2))**0.5
           b1= b1/norm
           b2= b2/norm
            print *, 'PARTAN b1 is',b1
            print *, 'PARTAN b2 is',b2
           return
           end

           SUBROUTINE FAIX
(x10,x20,x11,x21,x12,x22,b1,b2,x13,x23,y3)

           real x10,x20,x11,x21,x12,x22,b1,b2,x13,x23,y3
           real mo,r1,r2,r3,r,c,assu

           print *, 'Entering FAIX.'

c-------mo is the slope between point 0 and point 2 in x1x2
space
           mo=(x22-x20)/(x12-x10)
c-------r1 is the distance between point 0 and point 1
           r1=sqrt((x11-x10)**2 + (x21-x20)**2)
c-------r2 is the distance between point 1 and point 2
           r2=sqrt((x12-x11)**2 + (x22-x21)**2)
c-------r is the ratio of these two distances
           r=r1/r2
c-------r3 is the distance between point 0 and point 2
```

```
        r3=sqrt((x22-x20)**2+(x12-x10)**2)
        print *, 'mo=',mo
        print *, 'r1=',r1
        print *, 'r2=',r2
        print *, 'r3=',r3
        print *, 'r=',r
        write(7,*) 'r1,r2,r3,r,mo are '
        write(7,*) r1,r2,r3,r,mo

c------c is a parameter describing the eccentricity
        c=(r*mo+1)/(r*mo-mo**2)
c------assu times r3 is the assumed acceleration length
        assu=(c*((c-1)**2)*mo**2)/((1+(c**2)*mo**2)**2)
        print *,'c=',c
        write(7,*) 'c=',c
        write(7,*) 'assu=',assu
        print *,'assu=',assu
        x13=x12+assu*(x12-x10)
        x23=x22+assu*(x22-x20)
        call SIM (x13,x23,y3)
        return
        end


       SUBROUTINE RSM (x10,x20,ymm,x1f,x2f,yf)

        real x10,x20,x1p,x1n,x2p,x2n,rf
        real *y11,y1m,*y1h,ym1,ymm,*ymh,*yh1,yhm,*yhh
        real b0,b1,b2,b11,b22,b12
        real x1f,x2f,yf

        rf=0.5
        print *, 'Entering 3-k RSM.'

        x1p=x10+rf
        x1n=x10-rf
        x2p=x20+rf
        x2n=x20-rf

        call SIM (x1n,x2n,*y11)
        write(7,*) x1n,x2n,*y11

        call SIM (x1n,x20,y1m)
        write(7,*) x1n,x20,y1m

        call SIM (x1n,x2p,*y1h)
        write(7,*) x1n,x2p,*y1h

        call SIM (x10,x2n,ym1)
        write(7,*) x10,x2n,ym1

        write(7,*) x10,x20,ymm
        call SIM (x10,x2p,*ymh)
        write(7,*) x10,x2p,*ymh

        call SIM (x1p,x2n,*yhl)
        write(7,*) x1p,x2n,*yhl

        call SIM (x1p,x20,yhm)
        write(7,*) x1p,x20,yhm

        call SIM (x1p,x2p,*yhh)
        write(7,*) x1p,x2p,*yhh

30      b0=(*yll+ylm+*ylh+yml+ymm+*ymh+*yhl+yhm+*yhh)/9.0
        b1=(*yhl+yhm+*yhh-*yll-ylm-*ylh)/6.0
        b2=(*ylh+*ymh+*yhh-*yll-yml-*yhl)/6.0
        b11=(*yll+ylm+*ylh+*yhl+yhm+*yhh-
2*(yml+ymm+*ymh))/6.0
        b22=(*yll+yml+*yhl+*ylh+*ymh+*yhh-
2*(ylm+ymm+yhm))/6.0
        b12=(*yll+*yhh-*ylh-*yhl)/4.0
        print *, 'b0 =',b0
        print *, 'b1 =',b1
        print *, 'b2 =',b2
        print *, 'b11 =',b11
        print *, 'b22 =',b22
        print *, 'b12 =',b12
        write(7,*) 'b0,b1,b2,b11,b22,b12 ='
        write(7,*) b0,b1,b2,b11,b22,b12

        x2f=x20 + ((-b1*b12)+(2*b11*b2))/((b12*b12)-
(4*b11*b22))
        x1f=x10 + ((-b2*b12)+(2*b11*b1))/((b12*b12)-
(4*b11*b22))

        call SIM (x1f,x2f,yf)
        print *, x1f,x2f,yf
        write(7,*),'the final point is',x1f,x2f,yf

        if (4*b11*b22 .lt. b12*b12) then
         print *, '(x1f,x2f) is a saddle point.'
         write(7,*) '(x1f,x2f) is a saddle point.'
        else if( (b11.lt.0) .and.(b22.lt.0))then
         print *, '(x1f,x2f) is a maximum point.'
         write(7,*) '(x1f,x2f) is a maximum point.'
        else if( (b11.gt.0) .and.(b22.gt.0))then
         print *, '(x1f,x2f) is a minimum point.'
         write(7,*) '(x1f,x2f) is a minimum point.'
       end if
       return
       end
```

Problem: y=10-5(x1-3)**2 -15(x2+7)**2

```
 point 0=   10.00000        10.00000        -4570.000
   9.500000        9.500000        -4285.000
   9.500000        10.50000        -4795.000
   10.50000        9.500000        -4355.000
   10.50000        10.50000        -4865.000
 b0,b1,b2,b12 are
  -4575.000        -35.00000       -255.0000       0.0000000E+00
 slope in x1 direction is -0.1359800
 slope in x2 direction is -0.9907116
 point 1 =   9.864020        9.009289        -4070.034
 the new slope for x1 is  0.9907116
 the new slope for x2 is -0.1359800
   11.84544        8.737329        -4096.162
   7.882597        9.281249        -4085.385
 point 2 is   9.606606        9.044616        -4069.682
 partan slopes are -0.3807506     -0.9246778
 r1,r2,r3,r,mo are
  0.9999996        0.2598276        1.033208        3.848704
2.428566
 c=   3.000038
 assu=  2.4198353E-02
 faix point 3 is   9.597086        9.021497        -4057.933
   8.845104        7.195260        -3183.407
   7.322102        3.496549        -1736.066
   4.276097        -3.900873       -142.2109
  -1.815912        -18.69572       -2157.813
   1.230093        -11.29830       -282.7932
 partan/line point 3 is   2.999984       -6.999997
10.00000
 point 3 is   2.999984       -6.999997        10.00000
   2.499984        -7.499997        4.999965
   2.499984        -6.999997        8.749922
   2.499984        -6.499997        4.999879
   2.999984        -7.499997        6.250043
   2.999984        -6.999997        10.00000
   2.999984        -6.499997        6.249957
   3.499984        -7.499997        5.000122
   3.499984        -6.999997        8.750079
   3.499984        -6.499997        5.000036
 b0,b1,b2,b11,b22,b12 =
   6.666667        7.8837074E-05 -4.2915344E-05  -1.250000
-3.750000
  0.0000000E+00
 the final point is   2.999995       -7.000003        10.00000
 (x1f,x2f) is a maximum point.
```

Problem: $y = 100(x2 - x1^2)^2 + (1 - x1)^2$

```
 point 0=   5.000000        5.000000        -40016.00
   4.500000        4.500000        -24818.50
   4.500000        5.500000        -21768.50
   5.500000        4.500000        -66326.50
   5.500000        5.500000        -61276.50
 b0,b1,b2,b12 are
  -43547.50      -20254.00        2025.000        500.0000
 slope in x1 direction is -0.9950391
 slope in x2 direction is  9.9484265E-02
   3.009922        5.198968        -1494.510
 -0.9702346        5.596906        -2171.297
   1.019844        5.397937        -1899.092
 point 1 =   2.035566        5.296385        -133.9797
 the new slope for x1 is -9.9484265E-02
 the new slope for x2 is -0.9950391
   1.836598        3.306307        -1.145918
   1.438661      -0.6738498        -752.9236
   1.637629        1.316228        -186.8933
 point 2 is   1.853121        3.471574        -0.8685583
 partan slopes are -0.8995146      -0.4368905
 r1,r2,r3,r,mo are
   2.979213        1.833909        3.498419        1.624516
0.4856958
 c=   3.234417
 assu=  0.3167595
 faix point 3 is  0.8563176        2.987431        -508.1403
 point 3 is  0.8563176        2.987431        -508.1403
 point 0=  0.8563176        2.987431        -508.1403
  0.3563176        2.487431        -557.5954
  0.3563176        3.487431        -1129.689
   1.356318        2.487431        -42.09572
   1.356318        3.487431        -271.6623
 b0,b1,b2,b12 are
  -500.2607        343.3817        -200.4151        85.63177
 slope in x1 direction is  0.8636594
 slope in x2 direction is -0.5040758
   2.593637        1.979279        -2207.654
 -0.8710013        3.995582        -1051.278
 point 1 =  0.4109897        3.247347        -948.0226
 the new slope for x1 is -0.5040758
 the new slope for x2 is -0.8636594
 -0.5971618        1.520028        -137.9068
  -2.613465      -1.934610        -7695.244
  -1.605313      -0.2072911        -782.0326
 point 2 is -0.6546982        1.421448        -101.3068
 partan slopes are -0.6943644      -0.7196236
 r1,r2,r3,r,mo are
```

```
  0.5156290        2.114142         2.176114          0.2438951
1.036377
 c=   -1.525327
 assu= -0.8533999
 faix point 3 is  0.6348025         2.757858          -554.6810
 point 3 is  0.6348025         2.757858         -554.6810
 point 0=  0.6348025         2.757858         -554.6810
  0.6148025        2.737858         -557.0496
  0.6148025        2.777858         -576.0886
  0.6548025        2.737858         -533.3094
  0.6548025        2.777858         -551.9421
 b0,b1,b2,b12 are
  -554.5975        11.97165         -9.417908         0.1015625
 slope in x1 direction is  0.7859479
 slope in x2 direction is -0.6182927
   2.206698        1.521272         -1122.531
 -0.9370932        3.994443         -974.8847
 point 1 =  0.5173576         2.850250         -667.2105
 the new slope for x1 is -0.6182927
 the new slope for x2 is -0.7859479
 -0.7192279        1.278354         -60.87779
  -3.192399       -1.865437         -14554.33
  -1.955813       -0.2935417        -1705.145
 point 2 is -0.4340829         1.640818         -213.0004
 partan slopes are -0.6913621       -0.7225083
  -1.816807        0.1958017        -972.0287
  0.9486414        3.085835         -477.8248
 partan/line point 3 is -0.1003690         1.989566          -
393.0498
 point 3 is -0.1003690         1.989566         -393.0498
 point 0= -0.1003690         1.989566         -393.0498
 -0.1103690        1.979566         -388.2933
 -0.1103690        1.999566         -396.2028
 -9.0369038E-02    1.979566         -389.8306
 -9.0369038E-02    1.999566         -397.7562
 b0,b1,b2,b12 are
  -393.0207       -0.7726593        -3.958778         -4.0206909E-03
 slope in x1 direction is -0.1915617
 slope in x2 direction is -0.9814806
 -0.4834923       2.6605010E-02  -6.492270
  -1.249739       -3.899318         -2987.493
 -0.8666157       -1.936356         -725.6849
 point 1 = -0.4258662         0.3218570         -4.006978
 the new slope for x1 is  0.9814806
 the new slope for x2 is -0.1915617
   1.537095       -6.1266333E-02  -587.8308
  -2.388827        0.7049803        -2513.000
 point 2 is  0.1850708        0.2026166         -3.498802
 partan slopes are  0.1577361       -0.9874813
  0.5005429       -1.772346         -409.4576
 -0.1304014        2.177579         -468.0861
 partan/line point 3 is  0.1956938      0.1361127       -
1.603718
 point 3 is  0.1956938      0.1361127      -1.603718
 -0.3043062      -0.3638873      -22.53949
 -0.3043062       0.1361127      -1.890530
 -0.3043062       0.6361127      -31.24157
  0.1956938       0.3638873      -16.82205
  0.1956938       0.1361127      -1.603718
  0.1956938       0.6361127      -36.38538
  0.6956938      -0.3638873      -71.98218
  0.6956938       0.1361127      -12.19446
  0.6956938       0.6361127      -2.406738
 b0,b1,b2,b11,b22,b12 =
  -21.89624      -5.151964       6.885005      -5.438776
-25.00000
   19.56938
 the final point is  0.6847318     -2.5018558E-02  -24.49076
 (x1f,x2f) is a maximum point.
```

Bibliography

1. Faix, Lt Col Joseph, Class handout distributed in OPER 750, Response Surface Methodology. School of Engineering, Air Force Institute of Technology (AU), Wright- *Patterson AFB, OH, July 1987.

2. Fisher, Robert C. and Ziebur, Allen D. Integrated Algebra and Trigonometry (Second Edition). New Jersey: Prentice-Hall, Inc. 1967.

3. Hillier, Frederick S. and Lieberman, Gerald J. Introduction to Operations Research (Fourth Edition). Oakland, California: Holden-Day, Inc.,1986.

4. Himmelblau, David M. Applied Nonlinear Programming. New York: McGraw-Hill Book Co., 1972

5. Montgomery, Douglas C. Design and Analysis of Experiments (Second Edition). New York: John Wiley and Sons, 1984.

6. Myers, Raymond H. Response Surface Methodology. USA: 1976

7. Reklaitis, G.V. and others. Engineering Optimization Methods and Applications. New York: John Wiley and Sons, 1983.

8. Sparrow, 1Lt Kalla J. An Interactive Computer Package For Use With Simulation Models Which Performs Multidimensional Sensitivity Analysis By Employing The Techniques Of Response Surface Methodology. MS thesis, AFIT/Gor/OS/84D-12. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1984.

9. Wilde, Douglas J. Optimum Seeking Methods. New Jersey, Prentice-Hall, Inc.,1964.

10. Wilde, Douglas J. and Beightler, Charles S. Foundations of Optimization. New Jersey: Prentice-Hall, Inc., 1967

VITA

Major Billy G. Ploetner was born 10 June 1951 in Louisville, Kentucky. Growing up in a suburb of Louisville, he graduated from Fairdale High School in 1969. After working a year for General Electric, he attended the University of Louisville and received the degree of Bachelor of Arts in Mathematics in May, 1974. Upon graduation, he received a commission in the USAF through the ROTC program. He began active duty in November, 1974 at Laughlin AFB, Texas in the Undergraduate Pilot Training program. Having completed his pilot training and earned his wings in October, 1975, he served as a KC-135 co-pilot and aircraft commander at Ellsworth AFB, S. Dakota until May, 1981. He attended SOS in residence in summer 1981 and was assigned to the 4950th Test Wing at Wright-*Patterson AFB, Ohio. His duties consisted of research pilot, instructor pilot for EC-135 aircraft, and Chief, Wing Training branch until entering the School of Engineering, Air Force Institute of Technology in September, 1986 .

PERMANENT ADDRESS: 6506 Newcut Rd.

Fairdale, KY. 40118

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

AD-A190 593

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Distribution Unlimited; Approved for Public Release
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GST/ENS/88M-9	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION School of Engineering	6b OFFICE SYMBOL (If applicable) AFIT/ENS
7a NAME OF MONITORING ORGANIZATION	
6c ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433-6583	7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING/SPONSORING ORGANIZATION	8b OFFICE SYMBOL (If applicable)
9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Search: An Interactive Computer Program for Optimizing Two-Variable, Unconstrained Experiments or Simulation

12 PERSONAL AUTHOR(S)
Floetner, Billy G., Major, USAF

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
M.S. Thesis	FROM ______ TO ______	11 March 88	57

16 SUPPLEMENTARY NOTATION

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Line Search, Gradients, Response Surface Methodology, Direct Search, Optimization, Simulation, Partan, Operations Research, Factorial Design, Maximization

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Chairman: LtCol Joesph Faix
Assistant Professor of Operational Research

This thesis is a computer program that incorporates efficient techniques for solving optimization problems with unknown objective functions using steepest ascent and Partan methods.

[illegible] 21 Mar 88

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL LtCol Joesph Faix	22b TELEPHONE (Include Area Code) (513) 255-5533 — 22c OFFICE SYMBOL AFIT/ENS

DD Form 1473, JUN 86 — Previous editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED